http://www.mapleprimes.com/questions/202465-System-Of-Multivariable-Linear-Inequalities-12-Variables

```
> restart; interface(version);
```

Classic Worksheet Interface, Maple 18.00, Windows, Feb 10 2014, Build ID 922027

```
> sysRHS:= {1+F-J, 15+H-K, 18-J-K, 33-K-L, 34-H-J-L, -15-F+J+K+L, 16-F-H+J+K};
  vars:= convert(indets(sysRHS), list);
```

$$sysRHS := \{1 + F - J,\ 15 + H - K,\ 18 - J - K,\ 33 - K - L,\ 34 - H - J - L,\ -15 - F + J + K + L,\ 16 - F - H + J + K\}$$

$$vars := [F, H, J, K, L]$$

```
> Sys:={seq(1 <= q, q in sysRHS), # non-strict inequalities need for 'simplex'
        seq(1 <= q, q in vars) }; # positive integers
  #Sys:=convert(%, std);
```

$$Sys := \{0 \le F - J,\ 0 \le 14 + H - K,\ 0 \le 17 - J - K,\ 0 \le 32 - K - L,\ 0 \le 33 - H - J - L,\ 0 \le -16 - F + J + K + L,\ 0 \le 15 - F - H + J + K,\ 1 \le F,\ 1 \le H,\ 1 \le J,\ 1 \le K,\ 1 \le L\}$$

```
> # find hypercube for variables, use hash table for indexing

  for q in vars do
    Min[q]:= simplex[minimize]( q, Sys );
    Max[q]:= simplex[maximize]( q, Sys );
    Min[q]:= ceil( eval(q, Min[q]) );     Max[q]:= floor( eval(q, Max[q]) );
    print(Min[q] <= q, q <= Max[q]);
  end do: q:='q':
  ``;
  # http://www.mapleprimes.com/questions/202339-How-To-Use-Hash-Table-In-Maple
  `lower bounds` = [indices(Min, pairs)];
  `upper bounds` = [indices(Max, pairs)];
```

$$1 \le F,\ F \le 24$$

$$1 \le H,\ H \le 31$$

$$1 \le J,\ J \le 16$$

$$1 \le K,\ K \le 16$$

$$1 \le L,\ L \le 31$$

$$lower\ bounds = [L = 1, K = 1, H = 1, F = 1, J = 1]$$

$$upper\ bounds = [L = 31, K = 16, H = 31, F = 24, J = 16]$$

```
> # generate condition for coding

  tmp:=true:
  for q in sysRHS do
    tmp:= (tmp and 1 <= q);
  end do: q:='q':

  macro(theConditions=tmp):
  theConditions;
```

$$0 \le F - J \textbf{ and } 0 \le 14 + H - K \textbf{ and } 0 \le 17 - J - K \textbf{ and } 0 \le 32 - K - L \textbf{ and } 0 \le 33 - H - J - L \textbf{ and } 0 \le -16 - F + J + K + L \textbf{ and } 0 \le 15 - F - H + J + K$$

```
> # search the hypercube

  M:= 'M':
  n:= 0:
  st:= time():
  # hash table helps to prevent coding errors
  for F from Min['F'] to Max['F'] do
  for H from Min['H'] to Max['H'] do
  for J from Min['J'] to Max['J'] do
  for K from Min['K'] to Max['K'] do
  for L from Min['L'] to Max['L'] do
    if theConditions then
      n:= n+1:
      M[n]:= [F,H,J,K,L]:
    end if:

  od: od: od: od: od:
    F, H, J, K, L:= 'F', 'H', 'J', 'K', 'L': # clear variables

  `time needed`[sec]= time() - st;
  [seq(Max[q] - Min[q] +1, q in vars)]: convert(%, `*`): `# cases tested` = %;
  `# solutions`= n;
```

$$time\ needed_{sec} = 8.299$$

$$\#\ cases\ tested = 5904384$$

$$\#\ solutions = 150565$$

```
> # some random check
  randomize():
  ind:= RandomTools[Generate](integer(range=1..n));
  #ind:= 1;
  check:= [seq( vars[j]=M[ind][j], j=1 .. nops(vars))];
  eval(Sys, check); map(is, %);
```

$$ind := 21947$$

$$check := [F = 3, H = 19, J = 2, K = 12, L = 7]$$

$$\{0 \le 1, 0 \le 2, 0 \le 3, 0 \le 5, 0 \le 7, 0 \le 13, 0 \le 21, 1 \le 2, 1 \le 3, 1 \le 7, 1 \le 12, 1 \le 19\}$$

$$\{true\}$$

```
> # use table instead of long list
  MM:= rtable(1..n, M):
  MM[1..30]:   `first thirty solutions` = convert(%, list);
  MM[-30..-1]: `last thirty solutions` = convert(%, list);
```

$$first\ thirty\ solutions = [[1, 1, 1, 1, 15], [1, 1, 1, 1, 16], [1, 1, 1, 1, 17], [1, 1, 1, 1, 18], [1, 1, 1, 1, 19], [1, 1, 1, 1, 20], [1, 1, 1, 1, 21], [1, 1, 1, 1, 22], [1, 1, 1, 1, 23], [1, 1, 1, 1, 24], [1, 1, 1, 1, 25], [1, 1, 1, 1, 26], [1, 1, 1, 1, 27], [1, 1, 1, 1, 28], [1, 1, 1, 1, 29], [1, 1, 1, 1, 30], [1, 1, 1, 1, 31], [1, 1, 1, 2, 14], [1, 1, 1, 2, 15], [1, 1, 1, 2, 16], [1, 1, 1, 2, 17], [1, 1, 1, 2, 18], [1, 1, 1, 2, 19], [1, 1, 1, 2, 20], [1, 1, 1, 2, 21], [1, 1, 1, 2, 22], [1, 1, 1, 2, 23], [1, 1, 1, 2, 24], [1, 1, 1, 2, 25], [1, 1, 1, 2, 26]]$$

$$last\ thirty\ solutions = [[22, 5, 7, 10, 21], [22, 6, 6, 11, 21], [23, 1, 7, 7, 25], [23, 1, 7, 8, 24], [23, 1, 7, 9, 23], [23, 1, 7, 10, 22], [23, 1, 8, 7, 24], [23, 1, 8, 8, 23], [23, 1, 8, 8, 24], [23, 1, 8, 9, 22], [23, 1, 8, 9, 23], [23, 1, 9, 7, 23], [23, 1, 9, 8, 22], [23, 1, 9, 8, 23], [23, 1, 10, 7, 22], [23, 2, 7, 8, 24], [23, 2, 7, 9, 23], [23, 2, 7, 10, 22], [23, 2, 8, 8, 23], [23, 2, 8, 9, 22], [23, 2, 8, 9, 23], [23, 2, 9, 8, 22], [23, 3, 7, 9, 23], [23, 3, 7, 10, 22], [23, 3, 8, 9, 22], [23, 4, 7, 10, 22], [24, 1, 8, 8, 24], [24, 1, 8, 9, 23], [24, 1, 9, 8, 23], [24, 2, 8, 9, 23]]$$

```
>
```